Instructions.

- This is an individual assignment. No collaboration is allowed.
- Missing the deadline will cost you 100% marks.
- Submission will be only on MSTeams.
- This assignment makes 4% of your overall grade.
- Submit one PDF file. It should be named u2023XXX_a02.pdf where XXX are the last 3 digits of your registration number.

Questions (Max marks 100 – 20 each).

Task 1: Assuming you've setup the GPU coding environment in the last assignment, use the `cudaGetDeviceProperties()` function to get the information about your particular GPU in the `cudaDeviceProp` structure. For each field give a one-line explanation and its value for your GPU, e.g.,:

`devProp.clockRate`: represents GPU clock speed. Value = 2GHz.

Find out also the max Global Memory memory bandwidth and the peak compute performance of this GPU.

Task 2: Write a CPU program. It should multiply two, potentially non-square, matrice. The input data will be provided in a file. You can define the file structure (containing matrices' dimensions, elements) yourself and explain in your submission. The program when run will take max two filenames as argument. The first is the input data filename and the second (optional) argument is the output filename; if the output filename is absent it should write its output to stdout. The program will read input data (matrices) from the input file, store them in CPU memory, perform computations on them, write the result to output file in the same format structure you defined for input matrices. Compile run the code and push it to your github repo in the folder assignment02/task02. Use a build system i.e., Make, CMake, etc., for compilation. Write modular and parametrizable code. Activate compiler options which flag all warnings and report them. You program should compile with 0 warnings and 0 errors.

Task 3: Port the above program to GPU/CUDA. The program will read input data (matrices) from the input file, store them in CPU memory, copy them to GPU memory, perform operations on them, copy the result to CPU memory, and write it to output file in the same format structure you defined for input matrices. Compile run the code and push it to your github repo in the folder assignment02/task03. Use a build system i.e., Make, CMake, etc., for compilation. Write modular and parametrizable code. Activate compiler options which flag all warnings and report them. You program should compile with 0 warnings and 0 errors.

Calculate also the computational intensity for your program.

Task 4: Add code for measuring the computation time to both above programs. GPU time should include data transfer (to and from GPU) as well as computation time. Run both above programs for different matrix sizes and produce a graph of matrix size (x-axis) vs time (y-axis). Comment on what you see. Push it to your github repo in the folder assignment02/task04.

Task 5: Imrove the performance of your GPU program by using tiling. Try different tile sizes to determine the maximum tile size your GPU shared memory can accommodate, and then use that maximum tile size to re-run the program with the same matrix sizes as in Task 4. Plot a time versus matrix-size plot showing the performance of three methods (CPU, GPU-Naïve, GPU-Tiled) for different matrix sizes. Comment on what you see. Push it to your github repo in the folder assignment02/task05.

Calculate also the computational intensity for the tiled implementation and contrast it with the one in Task 3.

The End.